Tampereen ammattikorkeakoulu

Avoin lähdekoodi

Kulttuuri ja toimintamallit

Sander Kallio

OPINNÄYTETYÖ
Toukokuu 2020

Tieto- ja viestintätekniikka
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tieto- ja viestintätekniikka
Ohjelmistotekniikka

KALLIO, SANDER:
Avoin lähdekoodi
Kulttuuri ja toimintamallit

Opinnäytetyö 29 sivua, joista liitteitä 2 sivua
Toukokuu 2020

Opinnäytetyön tavoitteena oli tutustua avoimen lähdekoodin kulttuuriin ja toimintamalleihin. Kerätystä tiedosta rakennettiin kooste, josta selviää, mitä avoin lähdekoodi tarkoittaa ja miten sen ympärillä toimitaan. Avoimesta lähdekoodista löytyy paljon erilaista tietoa. Haasteena tiedon etsimisessä on virallisen tiedon löytäminen.

Avoimen lähdekoodin liike on saanut alkunsa Richard Stallmanista ja hänen pyrkimyksestään edistää lähdekoodin avoimuutta. Stallmanin aloittama GNU-projekti toimi avoimen lähdekoodin liikkeen kulmakivenä. Avoimen lähdekoodin projektissa lähdekoodi on käyttäjien luettavissa ja muokattavissa heidän tarpeidensa mukaan. Periaatteeseen kuuluu myös mahdollisuus käyttää ohjelmistoa haluamaansa tarkoitukseen. Alkuperäisen tai muokatun lähdekoodin levittäminen on myös sallittua.

Avoimen lähdekoodin kehitys tapahtuu pääosin vapaaehtoisvoimin. Projektiin osallistuminen kehittää osallistujan ohjelmointitaitoja ja kasvattaa sosiaalisia verkostoja. Ensimmäisen projektin löytäminen on kuitenkin haastavin osuus osallistumisesta. Usein projektin ylläpitäjät ovat dokumentoineet toivomansa osallistumistavat projektin tiedostoihin.

Omaa projektia aloittaessa tulee päättää, mitä projektilla tavoitellaan. Projektille luodaan selkeä dokumentaatio, jonka tarkoituksena on välittää oma visio projektista kiinnostuneille. Dokumentaatio sisältää ohjeet osallistumisesta ja projektin tavoitteista. Lisäksi projektille pitää valita lisenssi, jolla määritellään projektin avoimuus. Tärkein osa projektia on kuitenkin sen yhteisö.

Tutkimustyön tuloksena on rakennettu tämä opinnäytetyö, joka sisältää tiiviin kokoelman tietoa avoimesta lähdekoodista ja kuinka se toimii. Opinnäytetyön tavoitteiden saavuttamista on hankala todentaa, sillä aina löytyy uutta tietoa, mitä ei ole kirjattu. Opinnäytetyötä vois jatkokehittää lisäämällä projektien ylläpitämiseen liittyvää tietoa, kuten muutospyyntöjen käsittely prosessi.

Asiasanat: avoin lähdekoodi, Git, Github, projektinhallinta, versionhallinta, dokumentaatio

ABSTRACT

Tampere University of Applied Sciences
Information and communication technology
Software Engineering

KALLIO, SANDER:
Open Source
Culture and operating model

Bachelor's thesis 29 pages, appendices 2 pages
May 2020

The aim of the thesis was to get acquainted with the open-source culture and operating models. A compilation of information was studied to find out what an open-source means and how it is utilized. There are several sources of different information about open-source available, however, the difficulty is in finding official information.

The open-source movement originated from Richard Stallman and his efforts to promote open source transparency. The GNU project, launched by Stallman, served as the cornerstone for the open-source movement. In an open-source project, the source code is readable and customizable by users according to their needs. The principle also includes the ability to use the software for any purpose. Redistribution of the original or modified source code is also permitted.

The development of open source projects is mainly done voluntarily. Participating in the project develops the contributors programming skills and grows their social networks. However, finding the first project is the most challenging part of the process. Usually, project moderators have documented methods of participation in project files. The document also includes instructions for how to contribute to the project.

Starting a project requires setting the goals for the project. Clear documentation should be created to convey project vision to potential contributors. The documentation should include instructions on how to contribute to the project to achieve its goals. A license is required to define the openness of the project. However, the most important part of the project is its community.

The result of the research is a concise collection of information about open source and how it works. The thesis could be further developed with updated information and by adding information related to the maintenance of projects.

Key words: open source, Git, Github, project management, version control, documentation

SISÄLLYS

ERITYISSANASTO

ASCII-järjestelmästä	merkkikoodausstandardi sähköiselle viestinnälle.
BSD	permissiivisten lisenssien kokoelma
CI/CD	jatkuvan integraation ja toimittamisen malli
Git	hajautettu versionhallintajärjestelmä lähdekoodimuutosten seuraamiseksi ohjelmistokehityksen aikana.
GitHub	Git versionhallinta järjestelmän verkko tallennus palvelu
GNU	käyttöjärjestelmä ja kokoelma ilmaisia ohjelmistoja
Linux	Linux-ytimeen perustuva avoimen lähdekoodin käyttöjärjestelmät
ohjelmistovirhe	virheellisen toiminnan aiheuttava virhe ohjelmiston lähdekoodissa.
ohjelmointikieli	formaali kieli, joilla luodaan tietokoneelle suoritettavia algoritmeja.
Open Source Initiative (OSI)	avoimen lähdekoodin ohjelmisto käyttöä edistävä yhdistys.
pull request	versionhallintaan tehty yhdistämispyyntö erillisten koodien yhdistämiseen.

1 JOHDANTO

Avoimen lähdekoodin ohjelmistoja käytetään nykyään melkein kaikkialla. Ohjelmistojen lähdekoodit ovat vapaasti toisten luettavissa ja käytettävissä. Samalla tarjoten mahdollisuuden tuottavalle liiketoiminnalle. Tässä opinnäytetyössä käydään läpi, kuinka Avoimen Lähdekoodin liike sai alkunsa Richard Stallmanin toimiston tulostimesta. Ja kuinka siitä myöhemmin muodostui merkittävä osa nykypäivän ohjelmistokehitystä.

Idea lähti liikkeelle kiinnostuksesta Avoimeen Lähdekoodiin ja sen tuomaan kulttuurin. Materiaalia Avoimesta Lähdekoodista on paljon. Opinnäytetyön tarkoituksena on kerätä olennaiset tiedot useasta eri lähteestä ja rakentaa niiden pohjalta tiivis dokumentti. Dokumentin tavoitteena on esitellä Avoin Lähdekoodi käsitteenä ja opastaa alkuun sen toimintatavoissa.

Työssä haetaan vastauksia seuraaviin kysymyksiin:

- Mitä tarkoittaa avoin lähdekoodi ja mistä se tulee?
- Miten osallistuminen avoimen lähdekoodin projektiin tapahtuu?
- Miten avoimen lähdekoodin projektin luominen tapahtuu?

2 AVOIN LÄHDEKOODI

2.1 Lähdekoodi

Tietokoneohjelman toiminta perustuu sen lähdekoodiin. Ennen kuin ohjelmisto toimii pitää tietokonetta varten lähdekoodi kääntää tietokoneen ymmärtämään muotoon. Useimmiten käännös on tehty etukäteen kehittäjän toimesta. On myös olemassa ohjelmointikieliä, jotka käännetään lähdekoodia suorittaessa. Näitä ohjelmointikieliä kutsutaan tulkatuiksi kieliksi. JavaScript ja Python ovat tulkittuja kieliä, sillä niiden lähdekoodi käännetään suorittaessa. (Rouse, 2019)

Kun ohjelmiston lähdekoodi luettavissa. Kykenee osaava ohjelmoija laajentamaan ohjelmiston ominaisuuksia tai korjaamaan viallisia toiminnallisuuksia. Vapaasti luettava lähdekoodi helpottaa myös vastaavanlaisten ohjelmistojen kehitystä, toimimalla malli kappaleena. (Rouse, 2019)

2.2 Määritelmä

Määritelmä Avoin Lähdekoodi ei vain tarkoita pääsyä lähdekoodiin. Avoimen Lähdekoodin määritelmä on kehittynyt sen historian aikana. Open Source Initiative (OSI) on määritellyt Avoimen Lähdekoodin kymmenellä eri kriteerillä. Kriteerit ovat luettavissa englanniksi liitteessä 1.

1. Vapaa levitys ja uudelleenjakelu. Lisenssi ei saa rajoittaa yhtäkään osapuolta myymästä tai luovuttamasta ohjelmistoa osana koottua ohjelmistojakelua, joka sisältää ohjelmistoja useista eri lähteistä. Lisenssi ei myöskään saa vaatia lisenssimaksuja tai muita maksuja kyseisestä myynnistä.

2. Lähdekoodi. Ohjelman on sisällettävä lähdekoodi ja sen on sallittava jakelu lähdekoodina sekä käännetyssä muodossa. Jos lähdekoodia ei jaeta tuotteen kanssa, on oltava julkistettu tapa hankkia lähdekoodi kohtuullisin kustannuksin. Lähdekoodin on oltava ohjelmoijalle sopivassa muodossa.

Lähdekoodin luettavuuden tarkoituksellinen vaikeuttaminen ei ole sallittu. Välimuodot, kuten käännetty lähdekoodi eivät ole myöskään sallittuja

3. Johdetut teokset. Lisenssin on sallittava muutosten ja johdettujen teosten tekeminen. Lisenssin on myös sallittava muutettujen tai johdettujen teosten jakelu samoilla ehdoilla kuin alkuperäisen ohjelmiston lisenssi.

4. Alkuperäisen lähdekoodin eheys. Lisenssi saa rajoittaa lähdekoodin muokatun lähdekoodin levittämistä vain, jos lisenssi sallii korjaustiedostojen ja niiden lähdekoodin levittämisen. Lisenssi voi myös vaatia, ettei johdettuja teoksia saa levittää samalla nimellä tai versionumerolla kuin alkuperäinen ohjelmisto.

5. Ei yksilöiden tai ihmisryhmien syrjintää. Lisenssi ei saa asettaa yksilöitä tai ihmisryhmiä eriarvoiseen asemaan.

6. Ei käyttötarkoitusten syrjintää. Lisenssi ei saa rajoittaa ohjelmiston käyttötarkoituksia. Esimerkiksi, se ei saa rajoittaa ohjelmiston käyttöä liiketoiminnassa tai geneettisissä tutkimuksissa.

7. Lisenssin jakelu. Ohjelmistoon liittyvät rajoitukset ja oikeudet pitää olla samat kaikille, joille ohjelmistoa jaetaan. Ilman tarvetta erillisille lisensseille osalle käyttäjistä.

8. Lisenssi ei saa olla tuotekohtainen. Ohjelmistoon liittyvät oikeudet eivät saa olla riippuvaisia tietystä ohjelmistokokonaisuudesta. Jos ohjelmisto irrotetaan alkuperäisestä kokonaisuudesta ja käytetään tai jaetaan lisenssin mukaisesti. Pitää ohjelmiston käyttäjillä olla samat oikeudet kuin alkuperäisen kokonaisuuden käyttäjillä.

9. Lisenssi ei saa rajoittaa muita ohjelmistoja. Lisenssi ei saa asettaa ehtoja muille ohjelmistoille. Esimerkiksi lisenssi ei saa vaatia, ettei sitä levitä muiden kuin Avoimen Lähdekoodin ohjelmistojen kanssa.

10. Lisenssin pitää olla teknologia neutraali. Lisenssi pitää olla riippumaton teknisestä toteutuksesta. Lisenssi ei saa asettaa ehtoja yksittäisten teknologioiden tai käyttöliittymien suhteen.

OSI on myös perustellut kaikki kriteerinsä verkkosivuillaan (Open Source Initiative, n.d.). Perusteluissa kerrotaan syitä jokaiselle kriteerille. Esimerkiksi vaatimalla lisensseiltä ilmaista uudelleenjakelua. Voidaan vähentää houkutusta hylätä pitkän aikavälin suunnitelmat nopeiden hyötyjen saavuttamiseksi. Kriteerit myös asettavat vaatimuksia kulttuurin suhteen vaatimalla tasa-arvoa lisensseiltä.

2.3 Historia

Avoimen Lähdekoodin projektien tuottamia ohjelmistoja hyödynnetään melkein kaikkialla. Esimerkiksi älypuhelinten käyttöjärjestelmä Android tai Mozilla Firefox ovat Avoimen Lähdekoodin tuotoksia. Ennen tilanne oli kuitenkin toisin.

Richard Stallman ja tulostin

1970-luvulla Richard M. Stallman työskenteli MIT-yliopistossa. Siihen aikaan oli yleistä, että toimistossa oli vain yksi tulostin. Tulostimet yleensä jumittuivat, jolloin tulostusjono kasvoi, kunnes joku kävi korjaamassa tulostimen. Richard ja hänen työkaverinsa päättivät tämän takia kirjottaa ohjelman, joka ilmoittaa kaikille tulostusjonossa oleville, kun tulostin jumittuu. (Neary, 2018)

Vuonna 1989 toimistoon hankittiin uusi lasertulostin. Harmillisesti heidän kehittämää ohjelmaa ei voitu hyödyntää uuden tulostimen kanssa. Stallmanin kysyessä tulostimen valmistajalta tulostimen lähdekoodeja he kieltäytyivät. Hän kuitenkin kuuli, että eräällä toisen yliopiston tutkijalla olisi kyseiset lähdekoodit. Stallman kysyi häneltä tulostimen lähdekoodia, mutta yllätyksekseen sai kieltävän vastauksen. He olivat allekirjoittaneet salassapitovelvollisuuden tulostimen valmistajan kanssa. (Neary, 2018)

1970- ja 1980-lukujen vaihteessa oli yleistä, että ohjelmiston mukana luovutettiin ohjelmiston lähdekoodi. Samoihin aikoihin yliopistojen ohjelmistokehittäjät perus-

tivat yhä useammin uusia yrityksiä, jotka pyrkivät myymään ohjelmistojen lisenssejä. Tämä aiheutti salassapitovelvollisuuksien yleistymisen. Yritykset palkkasivat työntekijöitä yliopistoista työskentelemään yksityisissä kehityshankkeissa, joissa he eivät enää voineet osallistua aikaisempiin hankkeisiin tai jakaa osaamistaan toisten ohjelmistokehittäjien kanssa. (Neary, 2018)

GNU-Projekti

Syyskuussa 1983 Stallman ilmoitti GNU-projektin perustamisesta. Projektin tarkoituksena oli kopioida Unix-käyttöjärjestelmä, joka antaisi käyttäjälle täyden vapauden käyttöjärjestelmästä. Tammikuussa vuonna 1984 Stallman siirtyi työskentelemään projektin parissa täyspäiväisesti. Vuoden 1985 alussa hän julkaisi GNU Manifesto -dokumentin, jonka tarkoituksena oli kysyä tukea muilta GNU-järjestelmän kehittämiseen. Manifesti aloitti myös Free Software Foundation -nimisen organisaation, joka mahdollisti lahjoitusten antamisen projektin tukemiseksi. Dokumentti toimii myös perustamiskirjana Free software movement:lle. (Neary, 2018)

Ensimmäinen GLP-lisenssi

Vuoteen 1989 mennessä, kaikki Free Software Foundationin kirjoittamat ja julkaisemat ohjelmistot eivät sisältäneet yleisiä lisenssejä. Jokaiselle ohjelmistolle kirjoitettiin oma erillinen lisenssi, kunnes Unipress niminen yritys painosti Stallmania lopettamaan Emacsin toteutuksen kopioiden levittämisen. Stallman koki käyttäjien vapautta suojaavan lisenssin tärkeäksi. (Neary, 2018)

Ensimmäinen GNU General Public License julkaistiin vuonna 1989. Julkaisu samalla kiteytti copyleft -käsitteen. Termin tarkoituksena oli toimia vastakohtana *copyright* termille. Lisenssi mahdollisti lisenssin alaisten ohjelmistojen vapaan käyttämisen, muuttamisen ja jakamisen. Se vaati muuten koodin jakamisen ohjelmiston mukana, kun sen alaiseen ohjelmistoon tehtiin muutoksia. Lisenssin julkaiseminen ja Internetin tulo vuonna 1990, mahdollistivat hajautetun ja yhteystyöhön perustuvan Free software movementin kehitysmallin yleistymisen. (Neary, 2018)

Yritysten liittyminen

1990-luvulla VA Linuxin ja Redhatin siirtyminen pörssiin aiheuttivat voimakkaan kasvun ilmaisten ja avoimen lähdekoodin ohjelmistojen määrässä myös ammatillisessa käytössä. Molempien yritysten osakkaiden arvo kasvoi rähähdysmäisesti ensimmäisien päivien aikana pörssissä, samalla todistaen avoimen lähdekoodin siirtymistä kaupallisuuteen sekä sen yleistymisen. (Neary, 2018)

IBM ilmoitti vuonna 1999 tukevansa Linuxin kehitystä miljardilla dollarilla. IBM jalanjälkiä seuraten Sun Microsystems julkaisi lähdekoodin heidän toimisto-ohjelmistostansa, joiden pohjalta aloitettiin OpenOffice-projekti. Näiden ansiosta avoin lähdekoodi koki massiivisen käyttöönoton kasvun. Avoimen kehitysmallin omaksuminen johti Linuxin ja samalla avoimen lähdekoodin nousun hallitsevaan asemaan tämän päivän teknologiateollisuudessa. (Neary, 2018)

2.4 Hyödyt ja haitat

Avoimen lähdekoodin yksi suurimmista hyödyistä on sen hankintahinta, koska useimmiten perusohjelmisto on ilmainen. Ilmainen hankintahinta ei tarkoita, ettei kustannuksia olisi lainkaan. Esimerkiksi avoimen lähdekoodin ohjelmiston käyttöönottaneelle yritykselle voi syntyä kustannuksia sopivan henkilöstön kouluttamisesta. Jos myöhemmin ohjelmistosta löytyy toiminnallinen vika, yritys joutuu itse käsittelemään ongelman tai palkkaamaan ulkopuolisen korjaajan. Tämä johtuu siitä, että ohjelmistolla ei ole erillistä myyjää, jolta voitaisiin vaatia ohjelmiston korjausta. (Investintech.com, n.d.)

Yleisesti avoimen lähdekoodin ohjelmistoja pidetään hyvin luotettavina, sillä ohjelmistojen kehityksen parissa työskentelee usein monia alan ammattilaisia. Joskus niissä työskentelee jopa kymmeniä tai satoja vapaaehtoisia. Mahdollisuus lukea avointa lähdekoodia mahdollistaa koodiin jatkuvan kehityksen, silloinkin kun alkuperäiset kehittäjät ovat jo poistuneet projektista. Avoimuus myös mahdollistaa ohjelmiston muokkaamiseen omaan käyttötarkoitukseen sopivaksi. (Khalaf, 2017)

Ohjelmistojen kehitys on jatkuvaa, sillä kuka tahansa voi osallistua ohjelmiston kehittämiseen. Kun ohjelmistoa ylläpitävä yritys lakkauttaa toimintansa, loppuu myös tukipalvelut ja ohjelmistovirheiden korjaukset ohjelmistolta. Sen sijaan avoimen lähdekoodin ohjelmistoissa apua voi hakea ohjelmiston käyttäjäyhteisöltä. Haittapuolena yhteisölähtöiselle ohjelmistolle on juurikin sen yhteisö. Kun yhteisö menettää kiinnostuksensa ohjelmistoon, loppuu samalla sen kehitys ja tuki. Toisin sanoen avoimen lähdekoodin ohjelmiston kehitys voi loppua yhtäkkiä ilman erityistä syytä. (Investintech.com, n.d.)

Avoimen lähdekoodin ohjelmistoja pidetään turvallisempina yleisesti kaupallisiin ohjelmistoihin verrattuna. Ohjelmointivirheet ja muut ongelmat korjataan usein saman tien yhteisön jäsenten toimesta. Kaupallisilla ohjelmistoilla voi kestää viikkoja tai jopa kuukausia ongelmien korjaamisessa. Avoimuus tuo myös omat riskinsä tietoturvan kannalta. Koska lähdekoodi on avoimesti luettavissa ja muokattavissa, on mahdollista, että jollain kehittäjistä olisi haitallisia aikomuksia. Ei vaadita kuin yksi kehittäjä saamaan haittaohjelma ohjelmistoon ja syntyy ongelmia. Toisin on kaupallisissa ohjelmistoissa, joissa vain yrityksen kehittäjät pääsevät käsiksi lähdekoodiin. Tämä tuntuu turvalliselta, mutta se ei silti poista piilotetun takaoven mahdollisuutta. (Investintech.com, n.d.)

3 KEHITYS – Osallistuminen kehitykseen

3.1 Syitä osallistua avoimen lähdekoodin kehitykseen

Syitä osallistua avoimen lähdekoodin projekteihin on monia. Suurin syistä on kuitenkin oman osaamisen kehittäminen. ClemMakesAppsin kertoo artikkelissaan ”Benefits of Contributing to Open Source”, kuinka kokenutkin kehittäjä voi löytää projekteista uusi osa-alueita. Ja työskennellessään ennestään tuntemattoman osa-alueen parissa oppii tunnistamaan ohjelmistovirheitä ja kehittämään uusia toimivia ratkaisuja.

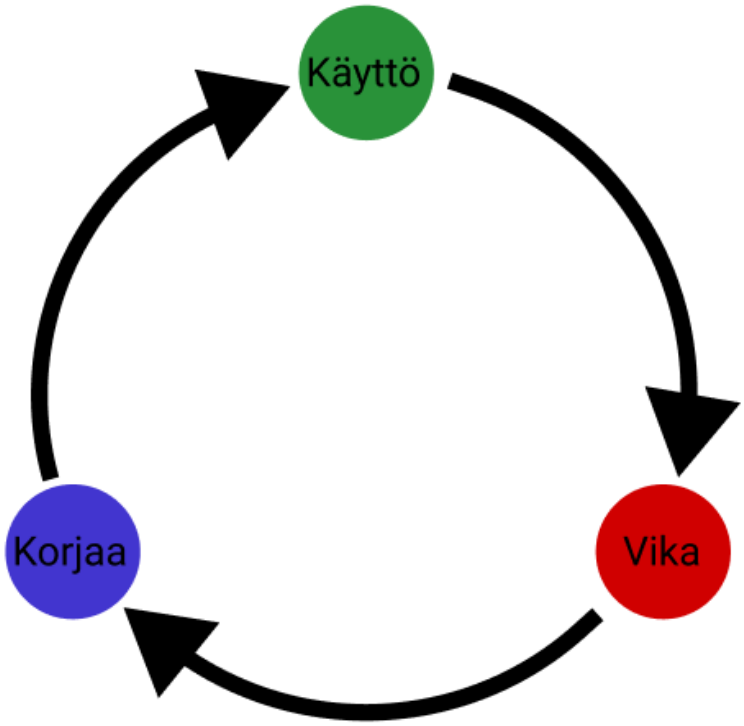
Koodin katselmointi tehostaa myös kehittymistä. Projekteissa katselmointeja suoritetaan pull request -pyynnöille, jolloin projektin ydinryhmä katselmoi yhdistettävän koodin. Katselmoidusta koodista palaute kehitys- ja korjauspyyntöjen muodossa. Prosessi ohjaa kehittäjiä noudattavaan hyviä toimintamalleja ja samalla kehittää kommunikaatiotaitoja. Ohjelmistokehittäjä saama palaute tehostaa myös kehittymistä kehittäjänä. Osallistumisesta hyötyä on projektin ylläpidolle. Muiden kehittäjien korjatessa pienempiä ongelmia, voi ydinryhmä keskittyä projektin hallinnointiin ja kriittisten ongelmien ratkaisuun. (ClemMakesApps, 2016)

Avoimen lähdekoodin projektiin osallistuessa omien taitojen lisäksi kehittyy maine kehittäjänä ja verkosto laajenee. Osallistumista projekteihin voi myös hyödyntää työnhaussa. Moni yritys arvostaa avoimen lähdekoodin kehitykseen osallistuneita ja samalla tehty työ toimii hyvänä esimerkkinä taidoista. Kasvanut verkosto auttaa myös työmarkkinoilla mahdollisesti tuomalla uusia työmahdollisuuksia.(ClemMakesApps, 2016)

3.2 Ensimäisen projektin löytäminen

Ensimmäisen projektin löytäminen on haastavin vaihe. Silloin vielä koko järjestelmä tuntuu uhkaavalta. Asiaa ei kuitenkaan tarvitse pelätä. Kannattaa lähestyä ensimmäistä projektia oman kiinnostuksen näkökulmasta. Sillä useimmiten osallistuminen projekteihin ei tarkoita sitoutumista, vaan enemmänkin jonkun

asian korjaamista. Rouvila kuvaa avoimen lähdekoodin kehitystä kierteenä (kuva 1). (Rouvila, 2020)



Kuva 1. Kehityksen kierre

Kehityskierre alkaa, kun käyttäjä löytää virheen projektin koodista. Löydöstään käyttäjä luo muutospyynnön projektin hallintasivulle. Jos käyttäjällä on osaamista korjata itse löytämänsä virhe, saattaa hän korjata sen itse ja luoda haaran yhdistämispyyntö ohjelmistovirhettä varten. Kun virhe on korjattu palaa hän takaisin alkuun käyttämään sovellusta. Jolloin osallistuminen projektiin tapahtuu oman kiinnostuksen/tarpeen kautta. (Rouvila, 2020)

Ensimmäisen projektin löytämiseen on myös muita keinoja. Suositeltu lähestymistapa on projektien sijaan etsiä muutospyyntöjä, jotka väistämättä ovat mukana jossain projektissa. Muutospyyntöjä hakemalla voi myös hyvin suodattaa tuloksia. Usein projektien ylläpito merkitsee aloittelijaystävälliset muutospyynnöt ”first-timers-only”-merkinnällä tai vastaavalla. Tällöin projektin ylläpito osaa myös odottaa ensikertalaista ja opastaa heitä oikeaan suuntaan. (Rouvila, 2020)

3.3 Osallistuminen projektiin

Useimmiten osallistuminen avoimen lähdekoodin projektiin käsitetään ohjelmoimisena. Kuitenkin osallistumisen voi hoitaa monella eri tapaa. Projektia voi tukea pelkästään auttamalla muita. Auttaminen voi tapahtua esimerkiksi vastaamalla Stack Overflow -foorumilla esitettyihin kysymyksiin tai auttamalla hallinnoimaan projektin muutospyyntöjä. Projektille on paljon apua, kuin joku tarjoutuu auttamaan muissakin tehtävissä, kuin pelkästään ohjelmoinnissa. Muut tavat ovat myös hyvä keino päästä sisälle projektiin ja tutustua yhteisön jäseniin. (Github, n.d.)

Ennen projektiin osallistumista pitää projektin dokumentaatiosta tarkistaa projektin lisenssi. Jos projektilla ei ole lisenssiä, se ei ole avoimen lähdekoodin projekti. Lisäksi projektilla olisi hyvä olla CONTRIBUTING-tiedosta. Tiedosto sisältää projektin ylläpidon ohjeet ja toiveet projektin osalta. Projekti voi olla myös hylätty, siksi on hyvä tarkistaa, milloin viimeksi projekti on edennyt. (Github, n.d.)

Yhteistyö on tärkeässä osassa avoimen lähdekoodin kehitystä. Siksi ennen muutospyynnön tai haaran yhdistämispyynnön tekemisestä on hyvä keskustella esimerkiksi projektin viestintäkanavalla, mahdollisten päällekkäisyyksien ja turhien pyyntöjen suhteen. Viestintäkanavalla muista antaa kysymyksillesi kontekstia ja muista tarkistaa, ettei kysymykseesi ole jo vastattu aikaisemmin. Hyviin tapoihin kuuluu myös pitää keskustelu avoimena ja kunnioittaa yhteisön päätöksiä. (Github, n.d.)

Kun muutospyyntö tai haaran yhdistämispyyntö on luotu ei prosessi ole vielä valmis. On mahdollista, ettei pyyntöön vastata. Jos vastausta ei tule viikon kuluttua, voi projektilta tiedustella voisiko joku katselmoida tuotoksen. Jos vieläkään et saa vastausta on mahdollista, että vastausta ei tule koskaan. Tällöin on suotavaa edetä seuraavaan projektiin tai etsiä toinen tapa auttaa projektia. (Github, n.d.)

Luotu muutospyyntö tai haaran yhdistämispyyntö ei välttämättä pääse suoraan mukaan projektiin, vaan siihen pyydetään muutoksia tai palautetta toteutuksen ideasta. Hyviin tapoihin kuuluu reagoida annettuun palautteeseen, eikä vain

unohtaa sitä. Jos jostain syystä esimerkiksi yhdistämispyynnön parissa työskentely ei onnistu, on suotavaa ilmoittaa tästä projektin ylläpidolle, jolloin toinen kiinnostunut voi ottaa yhdistämispyynnön työn alle. Kun yhdistämispyyntö liittyvät ongelmat on ratkaistu, siirtyy tehtävä ylläpidolle ja he yhdistävät sen projektiin. Tällöin osallistuminen projektiin kyseisen ongelman tai tehtävän osalta ohi ja osallistuminen avoimen lähdekoodin projektiin on onnistunut. (Github, n.d.)

4 OMA PROJEKTI – Projektin hallinnointi

4.1 Visio

Ennen projektin tuomista esille, on hyvä suunnitella mihin suuntaan projektia halutaan viedä. Vision tarkoitus on osoittaa yhteisölle, mitä projektilla tavoitellaan. Hyvä visio toimii koodaustyölle ohjaavana periaatteena ja vastaa yritys maailmassa usein esitettyihin kysymyksiin:

- Miksi projekti on olemassa?
- Mitkä ovat projektin tavoitteet?
- Mikä on ihanteellinen lopputilanne projektille?

Erillistä dokumenttia visiolle ei tarvitse tehdä, mutta se on hyvä pitää mielessä muuta dokumentaatiota kirjoittaessa. Sisällyttämällä vision ohjeistukseen, voidaan ohjata yhteisöä toivottuun suuntaan. Tällä tavalla vältytään monilta haasteilta jo alkutekijöissä. (Rouvila, 2020)

4.2 Versionhallinta

Versionhallintajärjestelmä seuraa ja tallentaa tiedostoissa tapahtuvia muutoksia, samalla mahdollistaen tiedoston palauttamisen aikaisempaan versioon. Versionhallintajärjestelmällä pystytään seuraamaan suurinta osaa erilaisista tiedostomuodoista. Järjestelmän käyttö on suotavaa tilanteissa, joissa voi olla tarve palata aikaisempaan versioon erillisen tiedoston kohdalla tai jopa koko projektin. Se mahdollistaa myös viimeisimmän muutoksen tekijän selvittämisen. (Github, n.d.)

Pääosin avoimen lähdekoodin kehityksessä käytetään Git versionhallintaa. Git käsittelee versiointia tilannekuvina, jolloin Git tallentaa aina commit-komennon yhteydessä hetkellisen version projektista. Jos tiedosto ei ole muuttunut sitä ei tallenneta. Git versionhallintaa voidaan käyttää täysin paikallisesti, joka mahdollistaa työskentelyn ilman verkkoyhteyttä. (Github, n.d.)

Git versionhallintaa käytetään yleisesti GitHub-palvelun kanssa. GitHub sisältää myös monia lisäominaisuuksia helpottamaan kehittäjän arkea. Usein sitä käytetään projektin hallinnollisiin tehtäviin, kuten muutospyyntöjen hallintaan versionhallinnan lisäksi. Github myös sisältää myös CI/CD mahdollisuuden. Pääsääntöisesti GitHubia kuitenkin käytetään Git versionhallinnan pilvitallennuspaikkana, josta kehittäjät voivat synkronoida projektin ohjelmistoa. (Brown, 2019)

4.3 Dokumentaatio

Dokumentaatio on tärkeä osa avoimen lähdekoodin projektia. Koodi alkaa useimmiten muistuttaa toisen henkilön tekemää koodia ajan kuluessa. Tällöin koodia on yleensä hankalaa tulkita ja koodissa toteutettuja ratkaisuja voi olla vaikea ymmärtää. Dokumentaatio mahdollistaa lähdekoodissa tehtyjen ratkaisujen perustelujen jakamisen. Holscher kuvaa dokumentaation tärkeyttä artikkelissaan Write The Docs sivustolla (Holscher, 2019).

1. Jos ihmiset eivät tiedä miksi projektisi on olemassa, he eivät käytä sitä.
2. Jos ihmiset eivät pysty selvittämään, kuinka asentaa ohjelmistosi, he eivät käytä sitä.
3. Jos ihmiset eivät pysty selvittämään, kuinka ohjelmistoasi käytetään, he eivät käytä sitä.

Jos projektia on tarkoitus jakaa muille, on dokumentaatio tärkeässä roolissa. Kaikki ihmiset eivät ole itse kiinnostuneita perehtymään ohjelmiston lähdekoodiin käyttääkseen ohjelmistoa. Ohjelmiston hyödyntämisestä kiinnostuneita käyttäjiä voisi olla enemmän, jos ohjelmistosta on olemassa hyvä dokumentaatio. (Holscher, 2019)

Yleisesti avoimen lähdekoodin projekti sisältää ”README”-, ”CONTRIBUTION”-, ”LICENSE”- ja ”CODE_OF_CONDUCT” -tiedostot. Dokumentaatioon tarkoitetut tiedostot yleensä kirjoitetaan isoilla kirjaimilla. Syy isoille kirjaimille johtaa ASCII-järjestelmästä, missä isot kirjaimet tulevat ennen pieniä. Tällöin ”README” tiedosto järjestäytyy ennen muita tiedostoja, mikä tekee siitä näkyvämmän kehittäjille. (Abdelhafith, 2015)

Dokumentaation sisältöön myös vaikuttaa vahvasti muiden vastaavien projektien malli. Esimerkiksi web-teknologioihin keskittyvät projektit useimmiten vastaavat toisiaan. On olemassa myös yhteisiäkin tapoja, joita useimmissa projekteissa noudatetaan, kuten README ja muut vastaavat tiedostot. Maailmalla on erilaisia tapoja dokumentaation laatimiseen, vastuu on kuitenkin viimeisenä kirjoittajalla. (Rouvila, 2020)

README -tiedosto on yksi tärkeimmistä projektin dokumenteista. Sen tehtävänä on toimia projektin etusivuna. Tiedostosta tekee tärkeän, koska se on ensimmäinen asia mitä ulkopuolinen näkee projektista. Usein vierailija tekee ensimmäiset johtopäätöksensä jo README tiedoston pohjalta. Tiedosto kuvaa myös hyvin millä tasolla projektista huolehditaan. Jos README tiedosto on huonolaatuinen tai sisältöinen, voidaan usein olettaa, että sama koskee muuta dokumentaatiota. (Rouvila, 2020)

README-tiedoston päätarkoitus on opastaa käyttäjiä ja mahdollisia uusia kehittäjiä. Tiedoston olisi hyvä kertoa mikä on projektin tai ohjelmiston tarkoitus. On myös hyvä tuoda esille mikä tekee projektista hyödyllisen. README-tiedoston ei ole pakko sisältää itse kaikkea tietoa, mutta sen olisi hyvä johdattaa oikeaan suuntaan esimerkiksi ohjaamalla CONTRIBUTION-tiedostoon. (Rouvila, 2020)

CONTRIBUTING-tiedosto löytyy myös useimmiten hyvän dokumentaation omaavan projektin juurihakemistosta. Sen tarkoituksen on opastaa projektiin osallistumisessa. Hyvät ohjeet yleensä sisältävät selkeät kuvaukset, miten projektiin voi osallistua ja miten osallistumisessa pääset alkuun. Tiedosto toimii tehokkaana tietolähteenä, ettei kaikille tarvitse opastaa erikseen. (Github, n.d.)

Alkuun tiedoston sisältö voi olla yksinkertainen. Sen on kuitenkin hyvä sisältää ohjeet esimerkiksi ohjelmistovirheestä ilmoittamiselle ja perustiedot, miten pystyttää kehitysympäristö. Myöhemmin voit laajentaa sisältöä esimerkiksi lisäämällä usein kysytyt kysymykset ja niiden vastaukset. Linkki tiedostoon on myös hyvä löytyä projektin README-tiedostosta. (Github, n.d.)

CODE OF CONDUCT -dokumentti perustaa projektiin osallistuvien käyttäytymistä koskevat odotukset, eli käyttäytymissäännöt. Säännöillä pyritään ylläpitämään positiivista ilmapiiriä. Ne suojaavat sekä osallistujia että projektin ylläpitäjiä mahdollisilta negatiivisilta asenteilta, joita saattaa nousta projektin edetessä esiin. Ennakoimalla voidaan vähentää projektin vetäjien rasitusta ja helpottaa hankalien tilanteiden ratkaisemista. Käyttäytymisääntöihin voidaan vedota myös mahdollisissa ongelmatilanteissa. (Github, n.d.)

Käyttäytymissääntöjen on hyvä ottaa kantaa projektin vetäjien odotusten lisäksi, myös ketä säännöt koskevat, mitä sääntöjen rikkomisesta seuraa ja kuinka niistä voidaan ilmoittaa. Tärkeää on, että ylläpitäjät valvovat käyttäytymissääntöjen noudattamista. Tällöin yhteisön luottamus käsittelyprosessiin säilyy. (Github, n.d.)

4.4 Lisenssit

Avoimen lähdekoodin projekteissa on tärkeä valita omalle tarkoitukselle sopiva lisenssi. Lisenssi on virallinen sopimus luojan ja käyttäjän välillä. Ilman lisenssiä ohjelmiston käyttäminen on kielletty, vaikka se olisikin vapaasti ladattavissa esim. GitHub-palvelusta. Valitsemalla avoimen lähdekoodin projekteille suunnatun lisenssin, kuten esim. MIT, GNU tai Apache tuot projektisi muiden saavutettavaksi. Lisenssin lisäys myös luo projektistasi avoimen lähdekoodin projektin. (Goldstein, 2019)

Lisenssejä on monenlaisia. Pääosin ne voidaan jakaa Copyleft- ja salliviin lisensseihin. Jako pohjautuu lisenssin vaatimuksiin ja rajoituksiin, jotka se asettaa käyttäjälle. Tekijänoikeus rajoittaa käyttö-, muokkaus- ja levitysoikeutta. Englanniksi termi on Copyright, josta pohjautuu termi Copyleft. Copyleft-lisenssillä tekijä julistaa tekijänoikeutensa projektille, jolloin hän mahdollistaa käytön, muokkaamisen ja jakamisen, kunhan lisenssin vaatimukset täyttyvät. Lisenssin alaisia komponentteja hyödyntävien ohjelmistojen pitää myös jakaa lähdekoodinsa muille luettavaksi (Goldstein, 2019)

Salliva lisenssi kuuluu Non-Copyleft -lisensseihin. Se mahdollistaa vapaan käytön ohjelmistolle. Salliva lisenssi pyrkii asettamaan mahdollisimman vähän vaatimuksia ja rajoitteita käyttäjälleen. Eli käyttäjä voi vapaasti käyttää, muokata ja jakaa ohjelmistoa. (Goldstein, 2019)

MIT-lisenssi on yksi sallivimmista lisensseitä. Sen tarkoituksena on olla lyhyt ja yksinkertainen salliva lisenssi, joka vaatii vain lisenssin sisällyttämisen projekteissa, joissa sen alaista sisältöä hyödynnetään. MIT-lisenssin yksinkertaisuus on luonut siitä yhden suosituimmista lisensseistä. Lisenssi on alun perin kehitetty MIT-yliopistossa. Vaikkakin lisenssi on suosittu, sillä ei ole tarkkaa dokumentaatiota sen historiasta. Sen arvioidaan saaneen alkunsa 1987, kun Project Athena haluttiin jakaa koulutuskäyttöön. Ongelmaksi nousi IBM:n päätös, ettei projektista tehtäisi täysin julkista. Jolloin MIT -yliopiston lakimiehet kirjoittivat ensimmäisen version MIT-lisenssistä. (Haff, 2019)

GPL-lisenssi on eniten käytetty avoimen lähdekoodin lisenssi. Lisenssi kuuluu ”Copyleft” kategoriaan, jolloin mikä tahansa ohjelmisto, joka hyödyntää GPL-lisenssin alaista sisältöä, pitää myös olla GPL-lisenssin alainen. Toisin sanoen ohjelmiston koodi pitää myös olla luettavissa kokonaisuudessa ja myös muutettavissa ja jaettavissa. (Shane, 2006)

Lisenssi sai alkunsa, kun Richard Stallman loi GPL-lisenssin GNU-ohjelmistoja varten. Lisenssin ensimmäinen versio julkaistiin vuonna 1989. Sen tavoitteena oli yhtenäistää kaikkien GNU-projektien lisenssit. Nykyjään kun käytetään GPL-lisenssiä, tarkoitetaan vuonna 1999 julkaistua 2.1 versiota. (Shane, 2006)

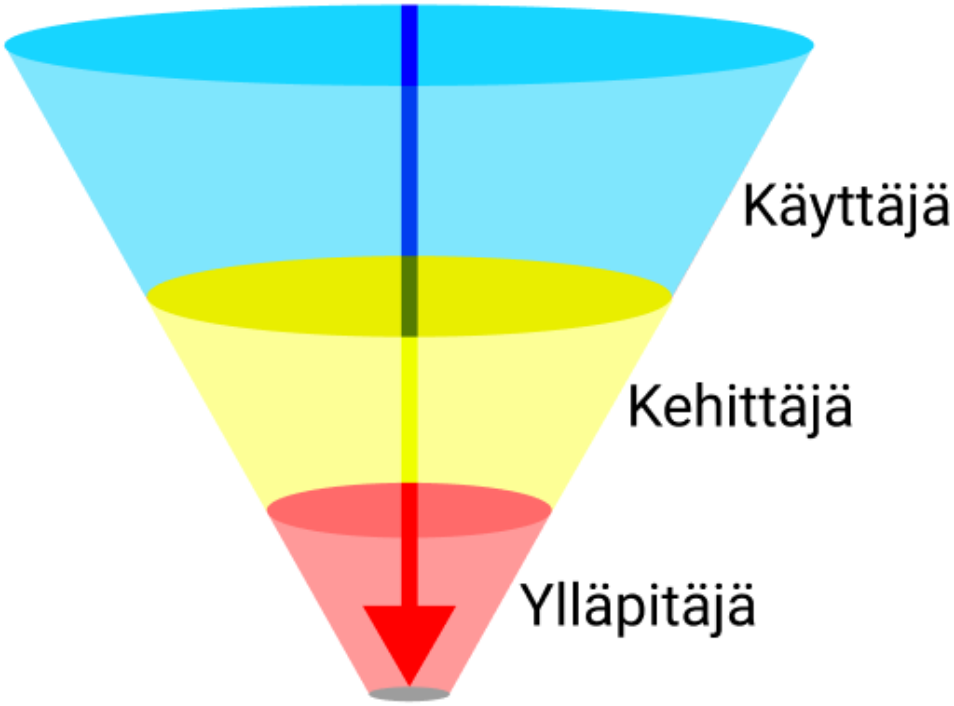
Apache-lisenssi on Apache Software Foundationin (ASF) julkaisema avoimen lähdekoodin lisenssi. Lisenssi mahdollistaa ohjelmiston vapaan käytön, muokkaamisen ja jakamisen. Apache lisenssiin on kirjoitettu vaatimukset, jotka pitää täyttää. Esimerkiksi kaikki muutokset, jotka on tehty alkuperäiseen ohjelmaan, pitää kirjata ylös. (Sass, n.d)

Lisenssi sai alkunsa 1995, kun Apache Group julkaisi ensimmäisen version lisenssistä. Vuonna 2000 lisenssistä poistettiin mainontalausekkeet ja samalla julkaistiin version 1.1. Nykyinen version 2.0 julkaistiin vuonna 2004, kun ASF päätti

mahdollistaa patenttioikeudet. Tällöin lisenssi ei enää ollut BSD-mallin mukainen, kuten se aikaisemmin oli ollut. (Sass, n.d)

4.5 Yhteisö

Mukava yhteisö on hyvä sijoitus projektin tulevaisuutta ja mainetta varten. Kun projektiin alkaa tulla ensimmäisiä ulkopuolisia kehittäjiä, on hyvä antaa heille rakentavaa palautetta ja tehdä heille helpoksi palata projektiin. Mike McQuaid jakaa avoimen lähdekoodin projektin yhteisön kolmeen eri rooliin, käyttäjä, kehittäjä ja ylläpitäjä. McQuaid myös kuvailee avoimen lähdekoodin yhteisöä suppilomaiseksi, jossa jokaisella roolilla on oma alueensa (kuva 2).



Kuva 2: Projektin yhteisösuppilo

Käyttäjiä ovat projektin ohjelmistoa hyödyntävät henkilöt, jotka eivät ole mukana kehityksessä, dokumentaatiossa tai ongelmien luokittelussa. Käyttäjät saattavat kirjoittaa muutos- tai korjauspyyntöjä, mutta useimmiten keskittyvät avun pyytämiseen kuin toistettaviin vikoihin.

Kehittäjät ovat se osa käyttäjiä, jotka osallistuvat kehitykseen ohjelmoimalla tai dokumentoimalla. Kehittäjät kuitenkin vaativat jonkun katselmoimaan tuotetun

koodin ja yhdistämään heidän muutoksensa projektiin, sillä heillä ei ole oikeuksia itse yhdistää omia muutoksiaan. Joissain tilanteissa tarvitsee kehittäjää opastaa Git:in ja GitHubin käytössä. Joillakin kehittäjillä voi olla hämmästyttävät ohjelmointitaidot, mutta ei ole kokemusta GitHubista.

Ylläpitäjät hoitavat projektin ylläpitoon liittyviä tehtäviä, kuten koodikatselmointi ja sen yhdistäminen projektiin. Ylläpidetyllä projektilla on aina vähintään yksi ylläpitäjä, useimmiten projektin aloittanut henkilö. Kun projektilla ei enää ole ylläpitäjiä, pidetään projektia hylättynä. Uusia ylläpitäjiä etsiessä on hyvä käydä läpi projektiin osallistuvia kehittäjiä. Useimmiten uusi ylläpitäjä kuitenkin joudutaan suostuttelemaan tehtävään.

McQuaid kertoo, kuinka yhteisösuppilo esiintyy hänen ylläpitämässä projektissa nimeltään Homebrew. Homebrew:llä on miljoonia käyttäjiä, tuhansia kehittäjiä ja kymmenen ylläpitäjää. Lukuja suhteuttamalla voidaan päätellä, ettei kymmenen käyttäjän projektille välttämättä löydy kymmeniä kehittäjiä. Käyttäjiä voidaan kannustaa etenemään suppilossa ohjaamalla heitä korjaamaan itse löytämiään vikoja. Näin ollen käyttäjiä rohkaistaan luomaan haaran yhdistämispyyntö avun pyytämisen sijaan. (McQuaid, 2018)

4.6 Rahoitus

Avoimen lähdekoodin projektin kehittämisestä voi myös olla rahallista hyötyä. Usein myös itse projektin pyörittäminenkin voi tuottaa kuluja, kuten servereiden ylläpito. Tämän takia kasvavien projektien on hyvä harkita mahdollisia rahoituskeinoja. Pääsääntöisesti rahoituskeinot voidaan jakaa kolmeen kategoriaan: lahjoitukset, tekninen tuki ja lisenssit.

Lahjoituksien kysyminen on selkeästi helpoin tapa saada rahoitusta. Se ei vaadi kehittäjää muuttamaan olemassa olevia käytäntöjä tai toimintaansa. Usea projekti on löytänyt rahoituksen hyödyntämällä Open Collective-palvelua lahjoitusten keräämisessä. Projektit ovat myös onnistuneet hyödyntämällä suoria maksuyhteyksiä, kuten Patreon, Gratipay ja Liberapay. Palvelut mahdollistavat lahjoitustoiminnon lisäämisen projektin README-tiedostoon. (Berry, 2017)

Teknisiä tukipalveluita tarjotaan useimmiten avoimen lähdekoodin projekteissa. Kehittäjät usein löytävät itsensä vastaamassa kysymyksiin, käsittelemässä ongelmia ja tarjoamassa teknistä tukea omasta hyvyydestään ja toiveesta projektin käytön kasvamisesta. Projektin suosion kasvaessa kasvaa myös tarve tekniselle tuelle. Jotkut kehittäjät päättävät rahoittaa toimintansa kirjoilla, hyödykkeillä ja maksullisilla tukipalveluilla. Useimmiten kehittäjät eivät kuitenkaan valitse tätä kanavaa rahoittamiselle. Todellisuudessa pienempien avoimen lähdekoodien projekteille ei ole edes hyötyä tarjota maksullista teknistä tukea. Myös jos projekti ei ole tarpeeksi suuri, ei myöskään ole tarvetta maksullisille koulutuksille tai kirjoille. (Berry, 2017)

Lisensoimalla osan ohjelmistoa on mahdollista veloittaa ohjelmiston käytöstä. Tämä on myös usein kaikkein houkuttelevin tapa avoimen lähdekoodin kannalta. Erinomainen esimerkki on Mike Perhamin Sidekiq. Sidekiq on ilmainen avoimen lähdekoodin ratkaisu, mutta tarjoamalla pro- ja yritysversion hanke tuottaa yli miljoona dollaria vuodessa. Kun projekti halutaan lisensoida, muuttuu projektista yritys. Usein kehittäjät eivät kuitenkaan ole liikemiehiä ja mieluummin haluavat keskittyä projektin kehittämiseen. (Berry, 2017)

5 POHDINTA

Projektin tavoitteena oli luoda tiivis tietopaketti, jonka pohjalta lukija saa käsityksen mitä Avoin Lähdekoodi tarkoittaa ja miten sen ympärillä toimitaan ja vaaditaan. Työ tehtiin Café Boardgamen toimeksiannosta. Tavoitteiden saavuttamista on hankala todentaa, sillä aina löytyy uutta tietoa, mitä ei ole kirjattu. Ajan myötä myös tämäkin opinnäytetyön tieto vanhenee Avoimen Lähdekoodin kulttuurin kehittyessä.

Tietoja selvitettäessä liikkeelle lähdettiin perustiedoilla. Mutta selvitystyön edetessä näkemys Avoin Lähdekoodin historiasta muuttui. Alun perin oli oletettu Avoimen lähdekoodin olleen jo pitkän aikaa vakiintunut termi, mutta toisin oli. Liikehän on vasta lähiaikana saavuttanut nykyisen asemansa. Kokonaisuudessaan selvitystyö toi monia uusia näkökulmia ja valmiuksia Avoimen Lähdekoodin suhteen.

Tiedon löytämisessä ilmeni myös omanlainen haasteensa. Avoimelle lähdekoodille löytyy paljon erilaisia materiaaleja. Lähteiden kirjosta sopivien aineistojen löytäminen esittäytyikin haastavammaksi kuin oli oletettu. Hyväksi lähteeksi monessa tapauksessa päätyi GitHubin ylläpitämä sivusto. Se sisälsi paljon selvitystyöhön liittyviä artikkeleita. Aineiston luotettavuuttakaan ei erillisesti tarvinnut selvittää, sillä GitHub toimii kuitenkin Avoimen Lähdekoodin ytimessä.

Opinnäytetyössä olisi voitu ottaa enemmän kantaa avoimen lähdekoodin projektien raha asioihin Suomessa. Tiedon löytäminen kuitenkin on haastavaa, sillä Suomen laki ei ota suoraa kantaa Avoimen lähdekoodin projektin rahallisiin asioihin. Vaan asia joudutaan päättelemään muiden pykälien nojalla. Linuxin voisi myös tuoda vahvemmin esiin. Vaikkakin se on ollut vahvasti mukana Avoimen Lähdekoodin liikkeen historiassa, ei nähnyt sitä tarpeelliseksi perehtyä siihen tarkemmin. Linuxhan ei ole ainoa iso projekti mikä on ollut käynnissä liikkeen alkuajoista saakka. Mukana on ollut myös Mozilla ja OpenLibre.

LÄHTEET

Abdelhafith, O. (13. Elokuu 2015). *README.md: History and Components*. Luettu 28.3.2020. Noudettu osoitteesta https://medium.com/@NSomar/readme-md-history-and-components-a365aff07f10

Berry, E. (4. Joulukuu 2017). *Why Funding Open Source is Hard*. Luettu 28.3.2020. Noudettu osoitteesta https://hackernoon.com/why-funding-open-source-is-hard-652b7055569d

Brown, K. (13. Marraskuu 2019). *What Is GitHub, and What Is It Used For?* Luettu 15.4.2020. Noudettu osoitteesta https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/

ClemMakesApps. (31. 8 2016). *Benefits of Contributing to Open Source*. Luettu 28.3.2020. Noudettu osoitteesta https://hackernoon.com/benefits-of-contributing-to-open-source-2c97b6f529e9

GIT. (ei pvm). *1.1 Getting Started - About Version Control*. Luettu 15.4.2020. Noudettu osoitteesta https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

GitHub. (ei pvm). *How to Contribute to Open Source*. Luettu 9.2.2020. Noudettu osoitteesta https://opensource.guide/how-to-contribute/

GitHub. (ei pvm). *Starting an Open Source Project*. Luettu 9.2.2020. Noudettu osoitteesta https://opensource.guide/starting-a-project/

GitHub. (ei pvm). *Why contribute to open source?* Luettu 9.2.2020. Noudettu osoitteesta https://opensource.guide/how-to-contribute/

GitHub. (ei pvm). *Why do I need a code of conduct?* Luettu 9.2.2020. Noudettu osoitteesta https://opensource.guide/code-of-conduct/

Goldstein, A. (24. 1 2019). *Open Source Licenses Explained*. Luettu 28.3.2020. Noudettu osoitteesta https://resources.whitesourcesoftware.com/blog-whitesource/open-source-licenses-explained

Haff, G. (26. 4 2019). *The mysterious history of the MIT License*. Luettu 15.4.2020. Noudettu osoitteesta https://opensource.com/article/19/4/history-mit-license

Holscher, E. (8. Lokakuu 2019). *A beginner's guide to writing documentation*. Luettu 28.3.2020. Noudettu osoitteesta

http://www.writethedocs.org/guide/writing/beginners-guide-to-docs/#why-write-docs

Investintech.com. (no date). *Pros & Cons of Open Source in Business*. Luettu 15.3.2020. Noudettu osoitteesta https://www.investintech.com/resources/blog/archives/7975-pros-cons-open-source-business.html

Khalaf, K. (11. 7 2017). *The Pros and Cons of Open Source Software*. Luettu 15.3.2020. Luettu 28.3.2020. Noudettu osoitteesta https://medium.com/4thought-studios/the-pros-and-cons-of-open-source-software-d498304f2a95

McQuaid, M. (14. Elokuu 2018). *The Open Source Contributor Funnel (or: Why People Don't Contribute To Your Open Source Project)*. Luettu 20.4.2020. Noudettu osoitteesta https://mikemcquaid.com/2018/08/14/the-open-source-contributor-funnel-why-people-dont-contribute-to-your-open-source-project/

Neary, D. (1. Helmikuu 2018). *6 pivotal moments in open source history*. Luettu 20.4.2020. Noudettu osoitteesta https://opensource.com/article/18/2/pivotal-moments-history-open-source

Open Source Initiative. (22. 3 2007). *The Open Source Definition*. Luettu 20.2.2020. Noudettu osoitteesta https://opensource.org/docs/osd

Open Source Initiative. (no date). *The Open Source Definition (Annotated)*. Luettu 20.2.2020. Noudettu osoitteesta https://opensource.org/osd-annotated

Rouse, M. (9 2019). *Source Code*. Luettu 28.3.2020. Noudettu osoitteesta https://searchapparchitecture.techtarget.com/definition/source-code

Rouvila, R. (7. 3 2020). CEO, Rare Agency. (S. Kallio, Haastattelija)

Sass, R. (no date). *Top 10 Apache License Questions Answered*. Luettu 16.2.2020. Noudettu osoitteesta https://resources.whitesourcesoftware.com/blog-whitesource/top-10-apache-license-questions-answered

Shane. (16. 10 2006). *History of GNU General Public License*. Luettu 27.2.2020. Noudettu osoitteesta https://www.eukhost.com/blog/webhosting/history-of-gnu-general-public-license/

LIITTEET

Liite 1. The Open Source Definition

Introduction

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost, preferably downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code

The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups
The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor
The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License
The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product
The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Restrict Other Software
The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

10. License Must Be Technology-Neutral
No provision of the license may be predicated on any individual technology or style of interface